Information Sciences 738 (2026) 123140

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Sonic: Fast and transferable data poisoning on clustering algorithms

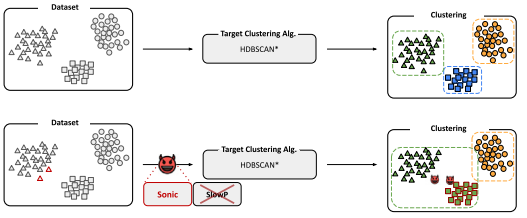
Francesco Villani [a], Dario Lazzaro [a,b], Antonio Emanuele Cinà [a,*], Matteo Dell'Amico [a], Battista Biggio [c], Fabio Roli [a,c]

[a] University of Genoa, Via Dodecaneso 35, Genoa, 16145, Italy
[b] Università Sapienza di Roma, Via Ariosto 25, Rome, 00185, Italy
[c] University of Cagliari, Via Marengo 3, Cagliari, 09100, Italy

HIGHLIGHTS

- Sonic is a fast genetic data poisoning attack against clustering algorithms for high-dimensional data.
- Sonic leverages incremental clustering algorithms to accelerate poisoning attacks for high-dimensional data.
- We investigate the trade-off between clustering approximation quality and attack speed.
- Data poisoning attacks carried out with Sonic transfer across different clustering algorithms.
- We analyze the empirical convergence of Sonic and provide an ablation study of its hyperparameters.

GRAPHICAL ABSTRACT



An illustrative example of a data poisoning attack against the clustering algorithm HDBSCAN* is shown. In the top row, we depict the scenario where the data are untainted, with HDBSCAN* correctly grouping the samples into three distinct clusters. In the bottom row, we depict a scenario where an attacker manipulates two red triangle data samples in the dataset to mislead the clustering algorithm. Specifically, the attack uses our fast and effective Sonic data poisoning method to perturb these samples, moving them from their original cluster (triangles) to a target cluster (squares), thereby maliciously influencing the clustering algorithm's results. As a result, this adversarial manipulation causes the HDBSCAN* algorithm to merge the blue and green samples into a single cluster, thus degrading the clustering performance.

ARTICLE INFO

Keywords:
Data poisoning
Adversarial machine learning
Clustering robustness
Transferability
Machine learning security
Unsupervised

ABSTRACT

Data poisoning attacks on clustering algorithms have received limited attention, with existing methods struggling to scale efficiently as dataset sizes and feature counts increase. These attacks typically require re-clustering the entire dataset multiple times to generate predictions and assess the attacker's objectives, significantly hindering their scalability. This paper addresses these limitations by proposing Sonic, a novel genetic data poisoning attack that leverages incremental and scalable clustering algorithms, e.g., FISHDBC, as surrogates to accelerate poisoning attacks against graph-based and density-based clustering methods, such as HDBSCAN. We empirically demonstrate the effectiveness and efficiency of Sonic in poisoning the target clustering algorithms. We then conduct a comprehensive analysis of the factors affecting the scalability and transferability

* Corresponding author.
Email address: antonio.cina@unige.it (A. Cinà).

https://doi.org/10.1016/j.ins.2026.123140
Received 21 December 2024; Received in revised form 19 January 2026; Accepted 20 January 2026
Available online 21 January 2026

of poisoning attacks against clustering algorithms, and we conclude by examining the robustness of hyperparameters in our attack strategy `Sonic`.

1. Introduction

Clustering algorithms are indispensable tools for organizing and deriving insights from vast amounts of unlabeled daily collected data [41]. They facilitate the identification of meaningful patterns and structures by grouping similar data points into distinct clusters (i.e., groups), enabling valuable decision-making processes in various industries [25]. In computer science, clustering algorithms are widely employed for tasks such as malware detection [39], anomaly detection [9], and topic modeling for identifying main themes in large text corpora [26]. Despite their broad adoption and practical relevance, the reliability and robustness of clustering algorithms have been increasingly questioned due to their demonstrated vulnerability to data poisoning attacks [11,12]. A data poisoning attack involves a malicious user (hereafter referred to as the attacker) who is capable of injecting or manipulating a small number of data points in the dataset with the objective of inducing inaccurate or misleading clustering outcomes. Such attacks can thus be used to undermine the effectiveness and integrity of clustering algorithms as attackers can distort clustering outcomes by merging distinct clusters or generating practically unusable results. Given the critical role of clustering algorithms in real-world systems, it is essential to develop fast, scalable procedures that systematically stress them and verify their resilience to adversarial perturbations. However, unlike the classification domain where much work has been developed [13], threats against unsupervised algorithms such as clustering are still little explored [14], along with the algorithms that can be used to test their robustness. Existing strategies [11,14] necessitate re-clustering the entire dataset multiple times to optimize their attack and evaluate the candidate solutions. This process can quickly become impractical due to the substantial time and resource demands, particularly as datasets grow in the number of features and volume of samples. Consequently, benchmarking the robustness of clustering algorithms with existing tools becomes complicated, making it nearly impossible and exposing them to potentially malicious users. Existing strategies [11,14] address data poisoning against clustering algorithms by iteratively searching for adversarial perturbations. To evaluate each candidate solution, these methods require re-running the target clustering algorithm on the entire dataset multiple times during the optimization process. As a consequence, the computational cost of the attack grows rapidly, making these approaches highly inefficient and impractical in settings where clustering algorithms are applied to large-scale and high-dimensional datasets. In particular, in such scenarios, the repeated execution of computationally expensive clustering procedures of existing attacks [14] leads to prohibitive time and resource requirements. As a result, benchmarking the robustness of clustering algorithms with current tools becomes extremely challenging and often impractical. This, in turn, prevents a systematic assessment of clustering robustness and leaves widely deployed clustering systems insufficiently evaluated.

In this paper, we address this open challenge by leveraging two key observations: (i) data poisoning attacks often target a small percentage of data to preserve attack stealthiness and practicability [8,12]; and (ii) the majority of clustering operations (e.g., distances between samples) remain valid for untainted data and do not require re-calculation. From these two observations, we derive `Sonic`, a genetic optimization strategy to stage data poisoning attacks against clustering algorithms.[1] `Sonic` leverages a faster and incremental surrogate clustering algorithm to accelerate optimization, facilitating robustness verification testing procedures. We show that generated attacks from `Sonic` benefit from the transferability property, i.e., generated adversarial noise can successfully mislead the original target clustering algorithm. As long as an adequate incremental surrogate algorithm exists, `Sonic` can accelerate attacks against *any* clustering algorithm. In this work, we focus on hierarchical and density-based algorithms because of *(i)* their high relevance[2] in practice [1,10,21,26]; *(ii)* their generally higher computational costs, which make attacks more expensive to evaluate.

Our experimental investigation provides compelling evidence of the remarkable performance of `Sonic`. We evaluate it on four benchmark datasets, including MNIST [31], FASHIONMNIST [47], CIFAR-10 [29], and the 20 Newsgroups dataset [30]. We compare its performance with state-of-the-art methodologies, showing that `Sonic` scales significantly better as both the number of samples and features in the dataset increase while preserving effectiveness, making it feasible to test the robustness of clustering algorithms on larger datasets. Lastly, we conduct a broader transferability analysis to evaluate the impact of `Sonic` perturbations on different clustering algorithms. We conclude the paper by discussing related work, the main contributions, and future directions to address the limitations of the proposed approach. We summarize our contributions as follows:

(i) We propose `Sonic`, a scalable poisoning attack for clustering algorithms that combines a genetic optimization strategy with incremental surrogate clustering to significantly accelerate the attack process, enabling effective black-box attacks and facilitating robustness verification procedures (Section 3);
(ii) We demonstrate that existing poisoning methods struggle to scale as the number of samples and features increases, whereas `Sonic` maintains strong attack effectiveness at significantly lower computational cost;
(iii) We analyze the transferability of our poisoning attack, showing that adversarial perturbations generated through our surrogate clustering algorithms reliably transfer to the black-box target clustering algorithms (Section 4.2).

[1] Code available at: https://github.com/francescovillani/sonic-poisoning-clustering

[2] The DBSCAN [22] paper, for example, won the SIGKDD Test of Time award in 2014 https://www.kdd.org/awards/view/2014-sikdd-test-of-time-award-winners.

The remainder of this paper is structured as follows: Section 2 introduces the clustering algorithms under study and the surrogate model utilized. Section 3 details our methodology, including the threat model considered in the paper, the problem statement, and the `Sonic` data poisoning algorithm. Section 4.1 describes the experimental setup, including the datasets and evaluation metrics adopted. Section 4.2 presents the experimental results, providing an in-depth analysis of `Sonic`'s qualitative results and inner workings. Section 5 reviews related work in unsupervised adversarial machine learning. We conclude the paper in Section 6 summarizing our contributions, key findings, and future directions to address the limitations of the proposed approach.

2. Clustering algorithms and incremental clustering

Data clustering is an unsupervised learning technique that creates groupings from unlabeled data; without access to label information, clustering algorithms are designed to uncover insights and identify shared patterns within the data. Clustering algorithms can be classified into various categories depending on how they operate; prominent among these are hierarchical algorithms (e.g., hierarchical clustering using different linkages [44,46]) and density-based algorithms (e.g., DBSCAN [22], OPTICS [3]). Specifically, density-based algorithms are popular because they recognize clusters of arbitrary shape and have the feature of distinguishing noise points: items that are not assigned to any cluster because they are in a low-density area [22]. Additionally, hybrid approaches, such as HDBSCAN* [7], combine features from multiple classes of algorithms. However, density-based clustering algorithms are known to be slower than other approaches such as, for example, the K-means algorithm. To this end, accelerated [33] and approximated [17,24] approaches have been developed to alleviate the problem, particularly in high-dimensional datasets.

In the following, we introduce the density-based, hierarchical, and hybrid clustering algorithms considered in this paper. Then, we present FISHDBC, an incremental algorithm used by the `Sonic` attack algorithm to accelerate the optimization process.

2.1. DBSCAN and HDBSCAN*

The seminal and most famous density-based clustering algorithm is DBSCAN [22]. In density-based algorithms, data points with enough similar neighbors are considered to belong to *dense areas* and will be assigned a cluster; the others are *noise* and will be assigned to no cluster. The similarity is generally expressed through a distance function; often, the samples to cluster are points in a Euclidean space, and the distance function chosen is Euclidean distance. In DBSCAN, a data point is considered in a dense area and called *core point* if at least *minPts* points are at a distance of at most ϵ from it, where *minPts* and ϵ are algorithm parameters. The algorithm then links in the same cluster the core points at a distance of at most ϵ from each other. Furthermore, DBSCAN has a concept of *border points*, which are not in a dense area but are within distance of ϵ from points in a dense area themselves. In an alternative implementation, DBSCAN* [7], border points are considered noise. The time complexity and scalability of DBSCAN vary depending on dataset characteristics and parameter choices [24]; in the worst case, it can grow up to $O(n^2)$.

HDBSCAN* [7] is a popular evolution of DBSCAN; it can be seen as a variation that supports hierarchical clustering and uses a heuristic to detect appropriate values of ϵ for different parts of the space to cluster. As a result, the algorithm requires one less hard-to-tune parameter; additionally, unlike DBSCAN, it can recognize clusters of different densities within a single dataset. Notably, we can observe parallels between HDBSCAN* and hierarchical clustering. By setting $minPts = 1$, HDBSCAN* can effectively mimic the behavior of the single-linkage clustering algorithm. Neither DBSCAN nor HDBSCAN* is incremental: if new elements are added to a dataset, the whole clustering has to be recomputed from scratch, requiring substantial resources.

Applications. Density-based clustering methods like HDBSCAN* and DBSCAN remain relevant in a plethora of real-world applications, even for high-dimensional data. They can be combined with techniques like UMAP to cluster textual data and perform tasks such as topic modeling [26], detect brand impersonation on social media [1], and identify fraudulent transactions [49]. Furthermore, DBSCAN variants cluster LiDAR point clouds in autonomous driving for obstacle detection [21] and boundary identification [10]. These applications demonstrate the capability of these clustering algorithms to handle complex, noisy, and high-dimensional data, making them valuable tools for unsupervised learning tasks in various domains.

2.2. FISHDBC: incremental density-based clustering

FISHDBC [17] is an algorithm that evolves HDBSCAN* in two directions: scalability for high-dimensional datasets or arbitrary distance/dissimilarity functions and support for incremental clustering. It leverages Hierarchical Navigable Small Worlds (HNSWs) [32], a data structure originally designed for approximate nearest-neighbor querying. FISHDBC piggybacks on the distance computations carried out by HNSWs and uses them to maintain its data structures. As a result, FISHDBC approximates HDBSCAN*, and by tuning the *ef* parameter, we can control the HNSW search cost [17]. Specifically, by raising it, we increase the computational cost, and we obtain a closer approximation of HDBSCAN*. Furthermore, an additional advantage of the algorithm's underlying structure is that FISHDBC is *incremental*: once the initial clustering is computed, new elements can be added to the dataset, and the clustering can be updated with minimal computational effort.

Lastly, similarly to HDBSCAN*, FISHDBC offers the possibility of configuring the algorithm to approximate DBSCAN* and single-linkage clustering. When attacking these algorithms, FISHDBC is a particularly effective surrogate algorithm closely mimicking the algorithm under study.

3. Sonic: fast data poisoning clustering

We present here Sonic, our genetic optimization strategy to rapidly stage poisoning attacks against clustering algorithms. In the following, we initially describe the threat model and then provide a formal overview of the proposed attack, its algorithmic implementation, and convergence properties.

3.1. Threat model

Let $D = \{\boldsymbol{x}_i \in \mathbb{R}^d\}_{i=1}^n$ and $C : \mathbb{R}^{n\times d} \to \mathbb{Z}^n$ denote respectively the dataset to be poisoned by the attacker, where n is the number of samples and d is the number of features, and the target clustering algorithm. We assume the attacker aims to stage a data poisoning attack [12] against the victim clustering algorithm C, leading to incorrect groupings [14]. To this end, we model a scenario where the attacker can only control a subset of samples $D_p \subseteq D$, with cardinality s. Realistically, the attacker has only a small percentage of control over the dataset because of access limitations or resource constraints [12,42], and does not influence remaining data D_c ($D = D_c \cup D_p$). We indicate with $D_p^\epsilon = \{\boldsymbol{x}_i + \boldsymbol{z}_i | \boldsymbol{x}_i \in D_p, \boldsymbol{z}_i \in \boldsymbol{\epsilon}\}$ the corresponding data after the injection of the poisoning noise $\boldsymbol{\epsilon}$ from the attacker to mislead data grouping from C. Furthermore, following the principles in [14] to make the attack more stealthy against human inspection of data, the attacker constrains the noise maximum intensity in $\boldsymbol{\epsilon}$ and looks for the perturbation that minimizes the number of tampered features in data samples $\boldsymbol{x}_i$. Unlike [14], we assume the attacker knows the internals of C and leverages such knowledge to make the attack faster on larger datasets.

3.2. Problem definition

We define $C(D)$ as the data grouping identified by C when applied to dataset D. The attacker aims to tamper with the portion of data under their control, D_p, by injecting an adversarial noise $\boldsymbol{\epsilon}$ to make the clustering algorithm C incorrectly group the data from the origin. This can be formalized as:

$$\begin{aligned} \min_{\boldsymbol{\epsilon}\in\Delta} \quad & \phi(C(D), C(D_c \cup D_p^\epsilon)) + \lambda\|\boldsymbol{\epsilon}\|_0, \\ \text{s.t.} \quad & \Delta = \{\boldsymbol{\epsilon} \in \mathbb{R}^{s\times d}, \|\boldsymbol{\epsilon}\|_\infty \le \delta\}, \end{aligned} \tag{1}$$

where ϕ is a similarity measure between clusterings [14] (e.g., AMI [37], ARI [27], or NMI [36]). ϕ quantifies how well the clusters in one partitioning correspond to the clusters in another. By injecting the poisoned data D_p^ϵ, the attacker wishes to lower the similarity between clustering outcomes C output on the untainted data D. The lower the score of ϕ, the higher the success rate for the attacker. Similarly to Cinà et al. [14], we use a penalty term $\lambda\|\boldsymbol{\epsilon}\|_0$ in the cost function to enforce the algorithm to search for poisoning samples with the minimal number of manipulated features. Lastly, the *adversarial attack space* Δ defines the space of poisoning perturbation masks that satisfy the maximum power constraints of the attacker.

3.3. Solution algorithm

We present our attacking algorithm, Sonic, for solving Eq. (1). Taking inspiration from [14], we configure Sonic as a genetic black-box attack. Our choice of a genetic algorithm is strongly motivated by the nature of clustering algorithms and by the constraints imposed by our threat model. Most clustering procedures are inherently non-differentiable [14], which makes gradient-based adversarial optimization infeasible in our setting. A derivative-free optimizer is therefore required, and genetic algorithms meet this requirement since they do not rely on assumptions of smoothness, convexity, or differentiability. We note that the specific choice of a black-box optimizer is orthogonal to our core contribution and could be replaced by alternative strategies, such as Bayesian optimization [6], which we leave for future work. The pseudocode of Sonic is reported in Algorithm 1.

At a high level, Sonic iteratively evaluates the quality of candidate adversarial perturbations and generates new solutions using genetic operators. The algorithm starts by initializing the adversarial noise to $\boldsymbol{\epsilon} = \mathbf{0}$ and creating the initial population set $\mathcal{E}$ (line 1). Next, it prepares a clustering state M by running the target clustering algorithm C on the clean portion of the dataset, denoted as D_c (line 2). Importantly, the state M represents the clustering structure obtained by C when considering *only* the clean data, which remains unchanged throughout the optimization. Using this prepared state M, Sonic computes a reference clustering partitioning $\mathcal{P}$ by updating a copy of M with the attacker-controlled data D_p and clustering the resulting untainted dataset $D = D_c \cup D_p$ (line 3). We refer to this operation as `update_cluster`. The partitioning $\mathcal{P}$ therefore represents the clustering outcome of C in the absence of any adversarial manipulation, i.e., the dataset D_p has not yet been tampered with by the adversary.

From this point, the Sonic algorithm employs an iterative genetic approach to generate new offspring solutions $\boldsymbol{\epsilon}$ to minimize the objective function specified in Eq. (1). In each iteration, Sonic performs the following steps. Sonic first constructs a poisoned dataset D_p^ϵ by injecting the current perturbation $\boldsymbol{\epsilon}^{(g)}$ into the attacker-controlled data D_p (line 5). It then applies the surrogate incremental clustering algorithm by updating the prepared state M with D_p^ϵ, yielding the poisoned clustering partitioning $\mathcal{P}^\epsilon$ (line 6). Importantly, this step constitutes a key improvement over prior work [14]. Rather than re-running the target clustering algorithm C on the entire dataset at each iteration, Sonic preserves the precomputed clustering state derived from the clean data M and incrementally updates it using only the poisoned samples. This significantly reduces the computational cost of the attack while maintaining high effectiveness. Sonic then evaluates the poisoning influence of $\boldsymbol{\epsilon}^{(g)}$ by comparing the untainted clustering $\mathcal{P}$ with the poisoned clustering $\mathcal{P}^\epsilon$, according to the objective defined in Eq. (1) (line 7). The population set $\mathcal{E}$ is subsequently updated with the new candidate (line 8), and the genetic operators `choice` (line 9), `crossover` (line 10), and `mutation` (line 11) are applied to generate the offspring perturbation for the next generation. Finally, after completing the optimization process, Sonic returns the adversarial noise that

Algorithm 1: `Sonic`: Fast and Transferable Clustering Poisoning Attack.

Input: $\mathcal{D}_c$, clean data; $\mathcal{D}_p$, data under attacker's control; C, target clustering algorithm; δ, maximum intensity constraint; ϕ, clustering similarity function; G, number of iterations; λ, Lagrangian multiplier.

Output: $\boldsymbol{\varepsilon}^\star$, optimal adversarial data poisoning noise.

1 $\boldsymbol{\varepsilon}^{(0)} \leftarrow \mathbf{0}, \quad \mathcal{E} \leftarrow \{\boldsymbol{\varepsilon}^{(0)}\}$

2 $M \leftarrow$ `prepare`$(C, \mathcal{D}_c)$ /* Model based on the clean data $\mathcal{D}_c$. */

3 $\mathcal{P} \leftarrow$ `update_cluster`$(M, \mathcal{D}_p)$ /* Clustering on $\mathcal{D} = \mathcal{D}_c \cup \mathcal{D}_p$. */

4 **for** g *in* $1, \ldots, G$ **do**

5 $\quad \mathcal{D}_p^{\boldsymbol{\varepsilon}} \leftarrow \{x_i + \boldsymbol{\varepsilon}_i^{(g)} \mid x_i \in \mathcal{D}_p,\ \boldsymbol{\varepsilon}_i^{(g)} \in \boldsymbol{\varepsilon}^{(g)}\}_{i=1}^{s}$ /* Poison $\mathcal{D}_p$. */

6 $\quad \mathcal{P}^{\boldsymbol{\varepsilon}} \leftarrow$ `update_cluster`$(M, \mathcal{D}_p^{\boldsymbol{\varepsilon}})$ /* Clustering on $\mathcal{D}_c \cup \mathcal{D}_p$. */

7 $\quad \Theta[\boldsymbol{\varepsilon}^{(g)}] \leftarrow \phi(\mathcal{P}, \mathcal{P}^{\boldsymbol{\varepsilon}}) + \lambda \|\boldsymbol{\varepsilon}^{(g)}\|_0$ /* Score candidate $\boldsymbol{\varepsilon}^{(g)}$. */

8 $\quad \mathcal{E} \leftarrow \mathcal{E} \cup \{\boldsymbol{\varepsilon}^{(g)}\}$ /* Increase population. */

9 $\quad \boldsymbol{\varepsilon}^{(g+1)} \leftarrow$ `choice`(Θ)

10 $\quad \boldsymbol{\varepsilon}^{(g+1)} \leftarrow$ `crossover` $(\boldsymbol{\varepsilon}^{(g)}, \boldsymbol{\varepsilon}^{(g+1)})$

11 $\quad \boldsymbol{\varepsilon}^{(g+1)} \leftarrow$ `mutation`$(\boldsymbol{\varepsilon}^{(g+1)}, \delta)$

12 **end**

13 **return** $\boldsymbol{\varepsilon}^\star \in \arg\min_{\boldsymbol{\varepsilon} \in \mathcal{E}} \Theta(\boldsymbol{\varepsilon})$

minimizes Eq. (1), denoted as $\boldsymbol{\varepsilon}^\star$. In the subsequent paragraphs, we expand on the implementation of the five key parts of `Sonic`, i.e., surrogate clustering (line 6), poisoning clustering evaluation (line 7), and the genetic operators (line 9-line 11).

Surrogate Clustering. The main limitation encountered in state-of-the-art data poisoning attacks [11,14] is their lack of scalability on large-scale data, as they require evaluating the target clustering algorithm C on the entire dataset for each optimization iteration. Density-based clustering algorithms (e.g., HDBSCAN* [7]) generally have non-trivial computational costs that depend on dataset characteristics, and this is exacerbated when, as in this case, clustering needs to be computed multiple times; in the worst case, computational complexity becomes $O(n^2)$ [24] , with n being the cardinality of the whole dataset $\mathcal{D}$. As a result, executing attacks against hierarchical and density-based algorithms, whether to mislead their outputs or assess their robustness using existing techniques [11,14], becomes infeasible for high-dimensional data.

However, we argue that the computational effort of these attacks can be substantially reduced by noting that data poisoning attacks typically target only a small percentage of the entire dataset [8,12]. It is thus reasonable to assume that most computations (e.g., distances between samples) remain valid for untainted data $\mathcal{D}_c$ and do not require significant adjustments or re-calculation. `Sonic` leverages these observations, implementing them effectively in Algorithm 1 to accelerate attacks against *any* clustering algorithm. Specifically, it uses incremental clustering algorithms (e.g., FISHDBC) as a *surrogate algorithm* to decrease the computational costs of poisoning attacks, leveraging the above observations. In line 2, `Sonic` prepares the clustering model M on the clean dataset not controlled by the attacker, $\mathcal{D}_c$. In this way, `Sonic` performs a partial computation based on the majority of input points that will not change and save its state. Afterward, the final clustering for untainted data is obtained in line 3 with the `update_cluster` function, which updates the state M with the data under the attacker's control, $\mathcal{D}_p$, which are the only points that change during the optimization process. For each data sample in $\mathcal{D}_p$, we add it to the saved state M and update the clustering. The same principle is employed in line 7. Rather than running the clustering algorithm C on the whole dataset, as done in [14], only a portion of the data is now considered, leading to less demanding updates and thus drastically reducing the computational effort of `Sonic`. Going into detail, as observed in later sections, we find that when the number of data points $s = \|D_p\|$ controlled by the attacker is small, the `Sonic` optimization procedure incurs drastically lower computational costs for the attacker. The time complexity for `update_cluster` is, in practice, dominated mainly by s, which we can expect to be much smaller than n [8,12]. In summary, the higher efficiency `Sonic` offers relies on minimizing the recomputation required for untainted data and on fast incremental updates that mainly scale with respect to the number of poisoned samples. Furthermore, we also demonstrate the significant effectiveness of `Sonic` in transferring the poisoning data, crafted with the incremental surrogate algorithm FISHDBC, on multiple target density-based and hierarchical clustering algorithms.

Poisoning Clustering Evaluation. The ϕ function in line 7 measures how the clusterings $\mathcal{P}$ of C on the untainted dataset $\mathcal{D}$ differs from the groupings identified in $\mathcal{P}^{\boldsymbol{\varepsilon}}$. Similarly to [14], we use the Adjusted Mutual Information (AMI) Score [37] as a similarity measure to evaluate the resulting clustering performance and to assess the impact of the adversarial perturbation. The AMI score between two clustering outcomes, $\mathcal{P}$ and $\mathcal{P}^{\boldsymbol{\varepsilon}}$, quantifies the similarity of the two cluster assignments by measuring the mutual information between them, adjusted for the possibility of agreement occurring by chance. It is formally defined as:

$$AMI(\mathcal{P}, \mathcal{P}^{\boldsymbol{\varepsilon}}) = \frac{MI(\mathcal{P}, \mathcal{P}^{\boldsymbol{\varepsilon}}) - \mathbb{E}[MI(\mathcal{P}, \mathcal{P}^{\boldsymbol{\varepsilon}})]}{\max\{H(\mathcal{P}), H(\mathcal{P}^{\boldsymbol{\varepsilon}})\} - \mathbb{E}[MI(\mathcal{P}, \mathcal{P}^{\boldsymbol{\varepsilon}})]} \tag{2}$$

where $MI(\mathcal{P}, \mathcal{P}^{\boldsymbol{\varepsilon}})$ represents the mutual information shared between the two clusterings. The term $\mathbb{E}[MI(\mathcal{P}, \mathcal{P}^{\boldsymbol{\varepsilon}})]$ denote the expected mutual information. The denominator's $\max\{H(\mathcal{P}), H(\mathcal{P}^{\boldsymbol{\varepsilon}})\}$ is the maximum of the entropies of the two clusterings, serving as an

upper bound for $MI(\mathcal{P}, \mathcal{P}^{\epsilon})$. AMI is equal to 1 when the two groupings $\mathcal{P}$ and $\mathcal{P}^{\epsilon}$ are identical, and 0 when they are independent of each other, i.e., they share no information. The AMI score makes no assumptions about the cluster structure and performs well even in the presence of unbalanced clusters, a plausible scenario when the attacker stages a targeted attack by moving samples only from one cluster towards others [14]. Nevertheless, compared to the clustering algorithms considered by Cinà et al. [14], density-based clustering algorithms, such as HDBSCAN*, support the notion of noise samples, i.e., points that do not fit well into any cluster and thus are considered anomalous. As a side effect of the AMI score, which looks at agreements between clustering outcomes, if two data points belonging to the same original cluster are marked as noise points after the poisoning process, the AMI score will be less affected. To mitigate this issue, during evaluation, we assign a unique label to each noise sample to ensure that the presence of noise does not artificially inflate the AMI score. This change ensures that noise points do not contribute positively to the AMI score, making it more sensitive to actual clustering performance changes.

Choice. The selection operator in a genetic algorithm serves to choose individuals from the population to contribute to the next generation, giving preference to those with better fitness scores [20]. In `Sonic` (line 7), the choice operator applies a softmax function to the attacker's objective function (see Eq. (1)) for each candidate in the population (i.e., $\Theta[\boldsymbol{\epsilon}_i]$), assigning each a probability of being chosen. Formally, the selection probability for a candidate $\boldsymbol{\epsilon}_i \in \mathcal{E}$ is inversely proportional to the value of Eq. (1), and is defined as:

$$p(\boldsymbol{\epsilon}_i) = \frac{\exp(-\Theta[\boldsymbol{\epsilon}_i])}{\sum_{\boldsymbol{\epsilon} \in \mathcal{E}} \exp(-\Theta[\boldsymbol{\epsilon}])} \tag{3}$$

Since the fitness score reflects both the attack's effectiveness and stealthiness, the selection process favors candidates that most effectively degrade the clustering results while maintaining minimal perturbation. Finally, `Sonic` employs an elitism strategy, maintaining a fixed population size and retaining only the most optimal offspring for the next generation.

Crossover. The crossover operation merges genetic information from two parent individuals to generate one or more offspring [20]. Crossover becomes fundamental in genetic algorithms for exploring new regions of the solution space and may reveal better performing candidates in future generations. Additionally, crossover helps to prevent the algorithm from prematurely converging on suboptimal solutions [20]. In `Sonic`, we follow the crossover technique proposed in [14]. Specifically, the crossover is executed by blending the current candidate with another selected through the choice operation, with their components being randomly swapped with a probability p_c (line 10).

Mutation. The mutation operation introduces random changes to individual genes within a candidate solution, thereby expanding the exploration of the solution space [20]. Like the crossover operation, mutation's randomness helps prevent the algorithm from getting trapped in local optima by enabling the discovery of potentially more effective solutions. In `Sonic`, a candidate perturbation is mutated with a probability p_m towards the nearest sample in the victim clusters, which encourages the merging of clusters. This strategy increases stealthiness by subtly shifting samples toward their neighbors and preserves effectiveness, as merging clusters significantly degrades clustering quality and, thus, the AMI score. Higher values of p_m enhance exploration, boosting the likelihood of finding diverse solutions, though this comes with increased stochasticity in the optimization process. Conversely, lower p_m values lead to a more focused search, potentially accelerating convergence but with a higher risk of missing better solutions. Additionally, following the strategy in [14], a zeroing operation with a probability p_z is used to eliminate perturbations on irrelevant features. This helps to refine the evolutionary process by concentrating the attack on more impactful changes. Combining mutation and zeroing ensures a balance between exploration and precision in crafting adversarial perturbations [14].

4. Experiments

This section presents an overview of the experimental process and its results. We begin by detailing the experimental setup (Section 4.1), including a detailed description of the datasets used and any preprocessing steps applied to the data. We also explain the hyperparameter selection process in our experiments and outline the evaluation metrics employed to assess performance. We then continue with the experimental results section (Section 4.2), where we first evaluate `Sonic`'s effectiveness in comparison to directly attacking HDBSCAN*, simulating attackers of varying strengths. Next, we examine `Sonic`'s scalability, emphasizing the benefits of using incremental clustering algorithms and testing different algorithm approximation levels to explore the trade-off between result quality and execution time for the attack. We then analyze the transferability of perturbations generated by `Sonic` by applying them to different density-based and hierarchy-based clustering algorithms. Finally, we evaluate `Sonic`'s empirical convergence properties and robustness to hyperparameter selection.

4.1. Experimental setup

Datasets. We employ four well-known datasets, which have also been used in recent related works [14,48,50]: MNIST [31], FASHIONMNIST [47], CIFAR-10 [29], and the 20 Newsgroups dataset [30]. MNIST and FASHIONMNIST feature 28×28 grayscale images of handwritten digits and fashion items, respectively. The CIFAR-10 dataset contains color images of size 32×32 pixels spanning 10 distinct classes. We preprocess CIFAR-10 data by taking the embedding representation from the last convolutional layers of a

pretrained ResNet-50 model trained on ImageNet.[3] The 20 Newsgroups text dataset is a collection of approximately $20,000$ newsgroup documents partitioned across 20 different topics. Following Albishre et al. [2], we employ a three-step preprocessing approach to transform the textual data into a suitable format for the clustering algorithms under investigation. Specifically, we begin by removing links, special characters, and numbers. Furthermore, we tokenize the text, remove stopwords,[4] and convert the remaining tokens to lowercase. The next step involves using a pre-trained sentence transformer [40] to convert the text documents into high-dimensional embeddings. Lastly, we apply Uniform Manifold Approximation and Projection (UMAP) [35] to reduce the dimensionality of the embeddings. We configure UMAP with 150 components and 15 neighbors and use cosine similarity as the distance metric for the dimensionality reduction process.

Problem Setup. Following the experimental setup of Cinà et al. [14], we focus part of our analysis on binary clustering problems where samples belonging to a victim cluster are perturbed towards a target cluster. We select pairs of labels with the highest clustering performance to ensure a challenging evaluation of the poisoning process. This choice allows us to assess the poisoning impact against a strong baseline, providing a fair measure of the attack's effectiveness. Specifically, we consider samples from digits 0 and 4 for MNIST, labels 'dress' and 'ankle boot' for FASHIONMNIST, and the 'automobile' and 'frog' classes for CIFAR-10, respectively as, victim and target clusters. We then follow the data extraction procedure from Biggio et al. [5] and pick the $2,000$ closest samples to the centroids of the two distributions. Complementary to our binary clustering analysis, we also consider a multi-cluster scenario. Specifically, for the 20 Newsgroups dataset, we select all samples belonging to four topics: 'comp.windows.x', 'rec.sport.baseball', 'soc.religion.christian', and 'sci.space', with a total of approximately $2,300$ samples. Our goal is to attack data in 'rec.sport.baseball' (victim cluster) and 'soc.religion.christian' (target cluster).

Evaluation Metrics. The main properties of `Sonic` we want to evaluate are its effectiveness in disrupting the clustering process and the time it takes to execute the attack. To measure the effectiveness of the attack, we employ the AMI score between the clusterings generated from the clean and poisoned data, as defined in Section 3. Furthermore, to establish a baseline for comparison, we execute the attack without its incremental features by attacking directly HDBSCAN*, as done by Cinà et al. [14]. This allows us to evaluate and compare both attacks' performance in terms of result quality and attack efficiency. All the experiments were run on a workstation equipped with an Intel(R) Xeon(R) Gold 5420 processor and 500GB of RAM.

Clustering Setup. Regarding the clustering algorithms,[5] we adjust each algorithm's influential hyperparameters by conducting a grid search and selecting the configuration that guarantees the best results. Specifically, we tune hyperparameters such as $minPts$, minimum cluster size, and ϵ to obtain the best baseline groupings, using default values when possible. The resulting clustering quality on clean data is reported in Table 1, confirming high AMI scores across all standard settings. McInnes et al. [34] provide an efficient Python implementation of HDBSCAN* which also supports computing DBSCAN*, the close relative of DBSCAN discussed in Section 2. The FISHDBC implementation[6] is based on the HDBSCAN* implementation referenced above [34].

Poisoning Setup. We run `Sonic` with the penalty term set to $\lambda = 0.1$ and probabilities $p_c = 0.85$, $p_m = 0.15$, and $p_z = 0.05$. The total number of iterations is set to 110. Moreover, we set the *ef* parameter of FISHDBC, used in the attack's optimization process, to 50. Further considerations and analysis on `Sonic`'s convergence properties and the influence of hyperparameters are provided in Section 4.2. Lastly, we conduct a baseline comparison using the poisoning strategy outlined in Cinà et al. [14], which involves using the target algorithm during the optimization process without any incremental approach. We refer to this attack as `SlowP`.

4.2. Experimental results

Effectiveness. We investigate whether `Sonic` can effectively disrupt the HDBSCAN* algorithm and compare its performance to the state-of-the-art method `SlowP` [14]. Our goal is to determine if `Sonic`, when combined with FISHDBC, can successfully transfer its data poisoning attack to the target algorithm. To this end, we apply `Sonic` and `SlowP` to the four datasets, gradually increasing

Table 1
Baseline clustering quality on clean data. We report the Adjusted Mutual Information (AMI) scores computed between the algorithms' predictions and the ground-truth labels across different datasets. The parameter ef controls the approximation level of FISHDBC.

Algorithm	Dataset			
	20 Newsgroups	MNIST	FASHIONMNIST	CIFAR-10
HDBSCAN*	0.810	0.959	1.000	0.765
FISHDBC ($ef = 50$)	0.810	0.994	1.000	0.811

[3] Model's weights have been sourced from the Torchvision https://github.com/pytorch/vision.
[4] Both the tokenizer and stopword remover are sourced from the nltk library.
[5] Sourced from Sklearn, Scipy, and HDBSCAN* [34] libraries.
[6] https://github.com/matteodellamico/flexible-clustering.

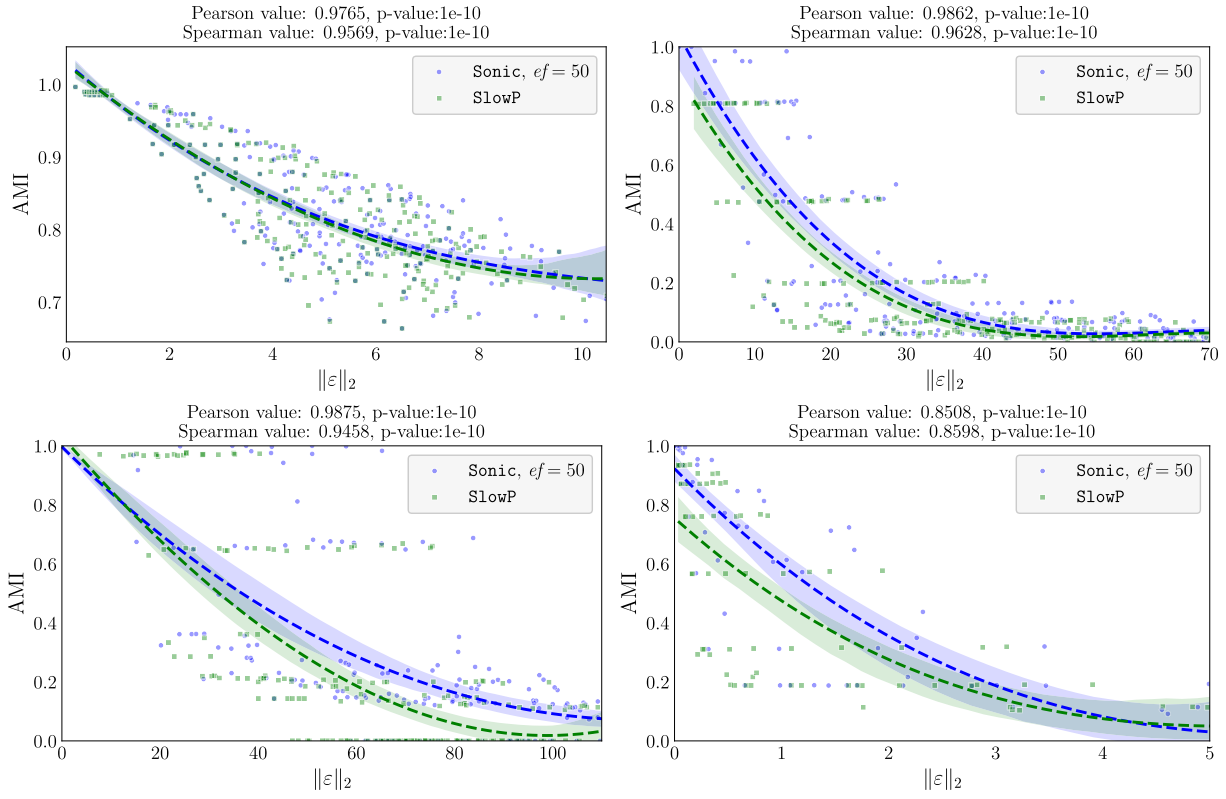


Fig. 1. Robustness analysis is conducted on four datasets: 20 Newsgroups (top-left), MNIST (top-right), FASHIONMNIST (bottom-left), and CIFAR-10 (bottom-right). Each point in the plots represents the outcome of an (s, δ)-experiment, where s ranges from 0.01 to 0.2 and δ ranges from 0.05 to 0.6. A regression line is included to illustrate the trend of our results. Additionally, Pearson and Spearman values are reported to indicate the statistical significance of the correlation between the effectiveness of `Sonic` and `SlowP`.

the perturbation size for each. Specifically, for all datasets except CIFAR-10, we execute the attack using 20 different values for s within the interval $[0.01, 0.2]$ and 12 values for δ within the interval $[0.05, 0.6]$, simulating attackers with varying strengths. For CIFAR-10, due to the higher computational effort required to attack HDBSCAN* directly, we sample 9 values each from $[0.001, 0.1]$ for both s and δ. Fig. 1 contains four scatterplots displaying the results of the attack procedures. Each point in the plots represents the outcome of a (s, δ)-experiment, and the regression lines indicate the trends in the results. Firstly, we confirm the attack's correctness, as results progressively worsen with increasing perturbation magnitude. Secondly, we identify a relationship between the two sets of results, represented in green (for `Sonic`) and blue (for `SlowP`), highlighted by the overlapping regression lines. Notably, we observe that the results for `Sonic` and `SlowP` are very similar to each other, and their correlation is further supported by the Pearson [38] and Spearman [45] correlation values displayed at the top of each plot. Ultimately, the results may differ across datasets or label configurations. On the 20 Newsgroups dataset, performance degradation is smooth, as both the δ and s constraints influence the poisoning problem in a balanced way. This suggests that perturbing few samples with higher magnitude has the same effect as perturbing a higher number of them with decreasing intensity. In contrast, for other datasets, one constraint often dominates the other, leading to significant shifts in performance only when the more restrictive constraint is altered. For example, in the MNIST (top-right plot) and FASHIONMNIST (bottom-left plot) datasets, the parameter δ is the more influential factor, leading to a significant drop in the AMI score of the final clustering results. In other words, for these datasets, it is more effective for the attacker to perturb a larger number of data samples, even if the perturbations are less visible, rather than tampering with a few samples but with a high δ.

We furthermore complete our effectiveness analysis by investigating the role of the *ef* parameter of FISHDBC in `Sonic`. We remark indeed that the *ef* parameter controls the HNSW search cost and the quality of the search process. The higher the *ef* parameter, the more precise the approximation of FISHDBC toward the target HDBSCAN* clustering algorithm. Fig. 2 presents regression lines for both `SlowP` and `Sonic`, configured with varying levels of FISHDBC approximation via the *ef* parameter. We also include the Pearson Correlation Coefficient (PCC) to assess the relationship between the results generated by different `Sonic` configurations and those produced by `SlowP`. As all the plots reveal, the regression lines generated with higher *ef* values align more closely with those of `SlowP`, reflecting the improved accuracy of FISHDBC in approximating HDBSCAN*. Nevertheless, as we will see in the "Efficiency" paragraph, higher accuracy levels affect the attack's execution time, highlighting the trade-off between precision and performance.

Efficiency. We investigate the runtime execution of `Sonic` compared to `SlowP` to demonstrate `Sonic`'s superior efficiency. Additionally, we examine the trade-off between `Sonic`'s efficiency and the approximation quality, which is controlled by the *ef* parameter. The

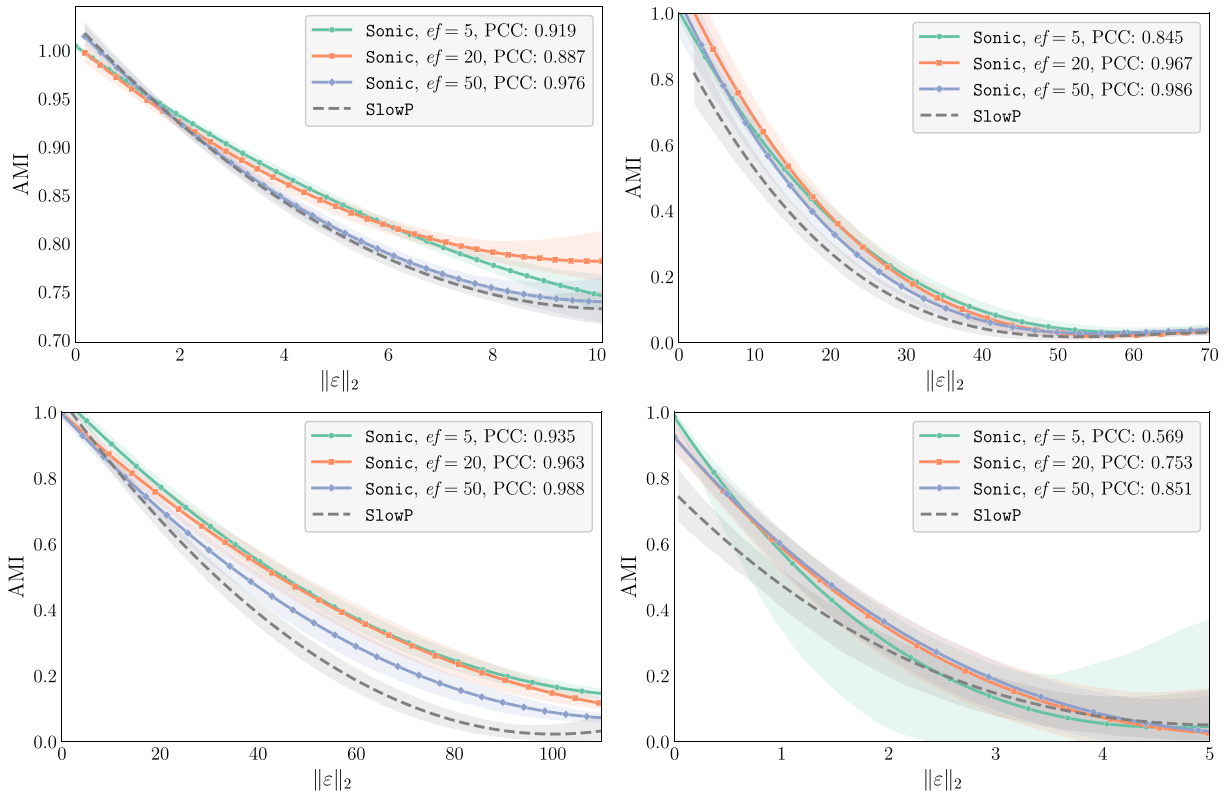


Fig. 2. Robustness analysis on four datasets: 20 Newsgroups (top-left), MNIST (top-right), FASHIONMNIST (bottom-left), and CIFAR-10 (bottom-right). We present results for `Sonic` at different FISHDBC approximation levels (*ef*), where lower *ef* values indicate more accurate approximations of HDBSCAN*. The Pearson Correlation Coefficient (PCC) is provided to show the correlation between the effectiveness of `SlowP` and `Sonic` across various *ef* levels.

results of these investigations are illustrated in Figs. 3 and 4. In more detail, similarly to the previous paragraph, Fig. 3 illustrates the execution time of the attack algorithms across the four datasets as the percentage of manipulated samples increases. The percentage of manipulated samples, as depicted in the plots, plays a key role in determining `Sonic`'s overall execution time. This is a consequence of the fact that `Sonic` exploits the incremental nature of FISHDBC during the optimization of the poisoning samples. Particularly, `Sonic` updates the clustering algorithm computational statements only on the attacked points and not on the whole dataset (line 6). Consequently, the more points considered in the optimization process, the more statements `Sonic` will need to update. When the number of manipulated data points is relatively low (a scenario commonly encountered in poisoning attacks [12]) `Sonic` delivers outstanding performance gains, markedly reducing its execution time. Conversely, for `SlowP`, the changing number of samples does not impact the final execution time; the algorithm maintains a consistent execution duration regardless of the number of perturbed data points. This is justified by the fact that `SlowP` updates the whole clustering outcomes at each attack iteration, hindering its practicability on large scale datasets. Going into the specifics of our results, we take into consideration, for example, the results for the MNIST dataset. Here `Sonic` with *ef* set to 50 and a poisoning ratio of 1%, is, on average, 27 times faster than `SlowP`, even when processing a modest set of $4,000$ samples. When the poisoning ratio increases to 10%, `Sonic` remains 8 to 10 times faster than `SlowP`. These speed-ups are even more pronounced with the larger CIFAR-10 dataset, where `Sonic` on average is 84 times faster with 1% poisoning ratio and 22 times faster when increasing the ratio to 10%. Finally, as shown in Fig. 3, these gaps can be adjusted by controlling the *ef* parameter. Specifically, reducing the search cost of FISHDBC's HNSW can indeed lower the quality of `Sonic`'s results; however, it considerably shortens its execution time, offering additional flexibility when evaluating clustering algorithms' robustness.

Scalability. To build on the time analysis discussed previously, we now conduct a supplementary analysis focusing on scalability of `Sonic` compared to `SlowP`. Specifically, we study the behavior of both attacks when dataset dimensionality (i.e., the number of samples and features) increases. For this analysis, we utilize synthetic Gaussian blob datasets,[7] varying in number of samples and number of dimensions. We first fix the number of samples at $4,000$, poisoning ratio to 10%, and vary the number of features within the range of $[100, 2.000]$. Subsequently, we set the number of features to 784 and explore how performance changes with varying sample sizes, ranging from 500 to 5.000. The results of our experiments are depicted in Fig. 4. The plots show that the `Sonic` consistently has a lower execution time than `SlowP`, performing similarly or worse only when the dataset has really

[7] The synthetic Gaussian blob datasets have been generated using Scikit-learn.

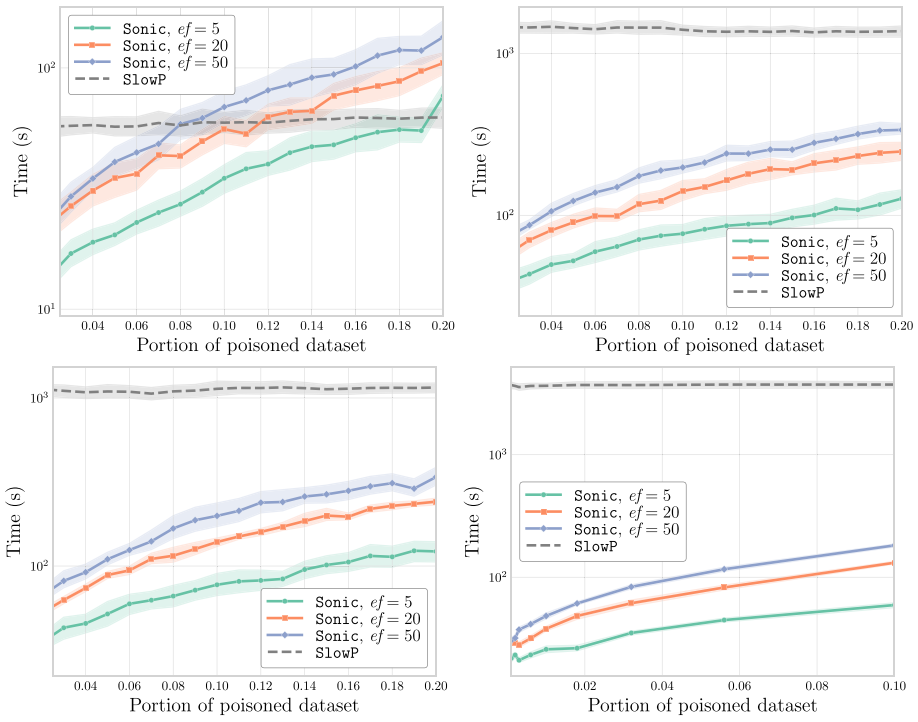


Fig. 3. Time analysis for `Sonic` at different approximation levels (*ef*) compared to `SlowP` on four datasets: 20 Newsgroups (top-left), MNIST (top-right), FASHIONMNIST (bottom-left), and CIFAR-10 (bottom-right). The x-axis represents the percentage of the dataset subjected to poisoning by the attacker, while the y-axis shows the total runtime of the attacks in seconds.

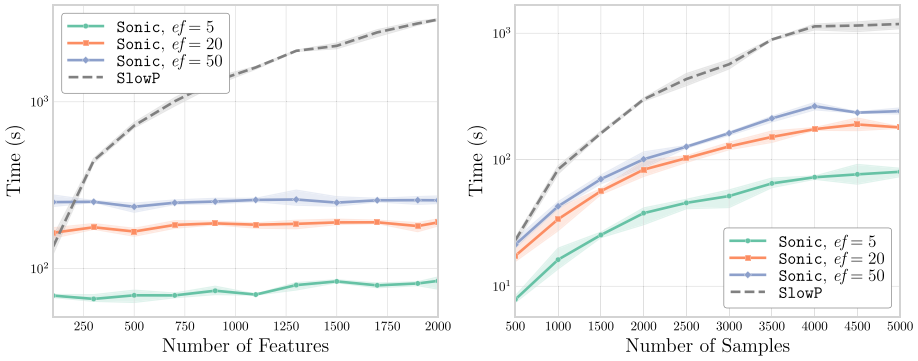


Fig. 4. Scalability analysis of `Sonic` at different approximation levels (*ef*) compared with `SlowP`. In the left plot, we increase the feature count of synthetic blob datasets, while in the right plot, we increase the number of samples, keeping the poisoning ratio fixed at 10%. The y-axis shows the total runtime of the attacks in seconds as the dataset dimensionality increases.

a few features (less than 200) and samples (less than 500). However, this phenomenon is not problematic since small datasets typically do not present scalability challenges. When tackling larger datasets, `Sonic` scales efficiently with the dimensionality of the data, maintaining consistent execution times as the number of features increases. In contrast, `SlowP` experiences significant scaling issues as the number of features grows; furthermore, while both approaches experience increased computational times with a growing number of samples, `Sonic` scales better than `SlowP`. Overall, `Sonic` leverages FISHDBC's underlying structure and its incremental design to more effectively manage large datasets, enabling the execution of poisoning attacks that would otherwise be unfeasible.

Transferability. We assess the *transferability* impact of `Sonic` generated poisoned samples against various clustering algorithms. This property, well-known for supervised learning applications in adversarial machine learning [18], is leveraged when the specific target model is unknown to the attacker. In this context, the attacker creates attacks using surrogate models and then tests them on the target model. However, we are the first to study this property in the context of data poisoning attacks against clustering algorithms. The goal is to evaluate how effective these attacks are across different clustering algorithms, providing insights into their performance when the

Table 2
Transferability results on the 20 Newsgroups dataset. Columns denote the source clustering algorithm used by `Sonic` to generate poisoned samples, while rows denote the target clustering algorithms. The table presents two scenarios: the left side shows a low-budget scenario, representing a more constrained problem with $\delta = 0.2$ and $s = 0.01$, where the attacker can manipulate fewer samples with lower magnitude; the right side shows a high-budget scenario, with $\delta = 0.5$ and $s = 0.1$, allowing the attacker to manipulate a greater number of samples and features. The values in brackets represent the difference in AMI between attacking the surrogate algorithm or directly the original one, i.e., `Sonic` and `SlowP`.

Target/Source	Low Budget: $\delta = 0.2$, $s = 0.01$			High Budget: $\delta = 0.5$, $s = 0.1$		
	$F_{HDBSCAN*}$	$F_{DBSCAN*}$	F_{H_S}	$F_{HDBSCAN*}$	$F_{DBSCAN*}$	F_{H_S}
HDBSCAN*	0.15 (−0.03)	0.38 (0.20)	0.33 (0.16)	0.00 (0.00)	0.08 (0.08)	0.10 (0.10)
DBSCAN*	0.07 (0.00)	0.07 (0.00)	0.07 (0.00)	0.07 (0.01)	0.07 (0.01)	0.07 (0.01)
H_S	0.04 (0.00)	0.04 (0.00)	0.04 (0.00)	0.04 (0.00)	0.04 (0.00)	0.04 (0.00)
H_C	1.00 (0.02)	1.00 (0.02)	1.00 (0.02)	0.69 (0.06)	0.90 (0.27)	0.94 (0.30)
H_A	0.99 (0.03)	1.00 (0.04)	1.00 (0.04)	0.69 (0.03)	0.85 (0.20)	0.92 (0.27)
H_W	1.00 (0.02)	1.00 (0.02)	1.00 (0.02)	0.69 (0.04)	0.91 (0.26)	0.94 (0.30)

Table 3
Transferability results on the 20 Newsgroups dataset. Columns denote the source clustering algorithm used by `Sonic` to generate poisoned samples, while rows denote the target clustering algorithms. The table presents two scenarios: the left side shows a low-budget scenario, representing a more constrained problem with $\delta = 0.2$ and $s = 0.05$, where the attacker can manipulate fewer samples with lower magnitude; the right side shows a high-budget scenario, with $\delta = 0.5$ and $s = 0.1$, allowing the attacker to manipulate a greater number of samples and features. The values in brackets represent the difference in AMI between attacking the surrogate algorithm or directly the original one, i.e., `Sonic` and `SlowP`.

Target/Source	Low Budget: $\delta = 0.2$, $s = 0.05$			High Budget: $\delta = 0.5$, $s = 0.1$		
	$F_{HDBSCAN*}$	$F_{DBSCAN*}$	F_{H_S}	$F_{HDBSCAN*}$	$F_{DBSCAN*}$	F_{H_S}
HDBSCAN*	0.68 (−0.01)	0.96 (0.27)	0.96 (0.27)	0.49 (0.00)	0.96 (0.47)	0.96 (0.47)
DBSCAN*	0.20 (−0.01)	0.21 (0.00)	0.21 (0.00)	0.20 (−0.02)	0.21 (0.00)	0.21 (0.00)
H_S	0.20 (−0.01)	0.21 (0.00)	0.21 (0.00)	0.20 (−0.02)	0.21 (0.00)	0.21 (0.00)
H_C	0.86 (0.13)	0.94 (0.20)	0.94 (0.20)	0.65 (−0.10)	0.87 (0.13)	0.87 (0.13)
H_A	0.77 (−0.01)	0.94 (0.15)	0.94 (0.15)	0.63 (0.06)	0.94 (0.37)	0.94 (0.37)
H_W	0.96 (0.24)	0.96 (0.24)	0.96 (0.24)	0.93 (0.35)	0.97 (0.39)	0.97 (0.39)

target model is not directly accessible. To achieve this, we generate poisoned data samples using `Sonic` with three distinct FISHDBC configurations to simulate the behavior of HDBSCAN* (denoted as $F_{HDBSCAN*}$), DBSCAN* (denoted as $F_{DBSCAN*}$), and hierarchical single linkage (denoted as F_{H_S}). Next, we inject these poisoned samples into the data and evaluate the performance of several target density-based and hierarchical clustering algorithms by running them on the poisoned dataset. Specifically, we test HDBSCAN*, DBSCAN*, and hierarchical linkage methods, including single linkage (H_S), average linkage (H_A), complete linkage (H_C), and Ward's linkage (H_W). We explore two scenarios: a constrained scenario with low δ and s values and a less constrained scenario with higher δ and s values. Tables 2 and 3 present the transferability experiments conducted on the MNIST and 20 Newsgroups datasets. The tables display the AMI scores obtained by testing the various clustering algorithms (rows) on poisoned datasets generated using the different configurations of `Sonic` (columns). The values in brackets represent the difference in AMI values between attacking the surrogate algorithm or directly the original, i.e., `Sonic` and `SlowP`. As can be seen in both tables, the poisoned samples generated using `Sonic` configuration with $F_{HDBSCAN*}$ demonstrate the best overall transferability to all other algorithms under study. Additionally, the various hierarchical linkages, except for H_S, show greater robustness to the perturbations generated by both `Sonic` and `SlowP`. Nonetheless, `Sonic` configured with $F_{HDBSCAN*}$ consistently produces results closest to the baselines.

Convergence. We now examine the empirical convergence of `Sonic`, focusing on the number of iterations required for the algorithm to reach an optimum. Fig. 5 illustrates various constrained scenarios across two datasets, 20 Newsgroups and MNIST, highlighting the best fitness value achieved at each step of the iterative process. Our results demonstrate that `Sonic` converges within a relatively small number of iterations, consistently improving the best solution throughout the optimization. Although different constraints affect the optimal solution, all experimental configurations converge in approximately 110 iterations.

Hyperparameters Study. Lastly, we conduct a hyperparameter study on the mutation probability parameter p_m and the zero mutation probability p_z of `Sonic`, with the crossover probability fixed at $p_c = 0.85$. The attack is performed with different hyperparameter configurations, where p_m ranges from $[0.01, 0.2]$ and p_z ranges from $[0, 1]$. The results obtained by running `Sonic` with these configurations are shown in Fig. 6. We perform the study in two different settings: the first on the multi-cluster problem using the 20 Newsgroups dataset and the second on the two-cluster problem using the MNIST dataset. The heatmaps reveal that high values of the zero mutation probability p_z significantly reduce noise and diminish the attack's effectiveness. Conversely, setting $p_z = 0$ results in more detectable perturbations and overall poorer performance. As noted in Cinà et al. [14], maintaining low non-zero values of p_z facilitates the optimization process in discovering better solutions while keeping the perturbation stealthier. The mutation probability parameter p_m shows resilience across different settings; nevertheless, very high p_m values introduce greater stochasticity, which may

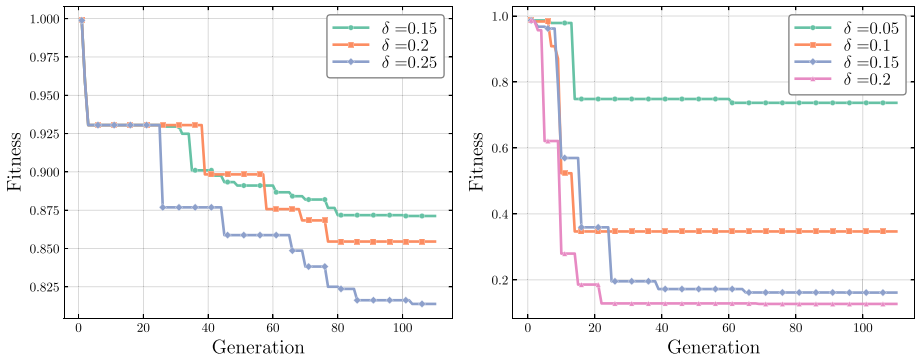


Fig. 5. Convergence curves of `Sonic` showing the best fitness value at each iteration. The left plot illustrates an example of convergence on the 20 Newsgroups dataset, with δ in $[0.15, 0.25]$, while the right plot shows the convergence on the FASHIONMNIST dataset, with δ in $[0.05, 0.2]$. The attacks have been run for 110 iterations each, fixing the poisoning ratio to 15%.

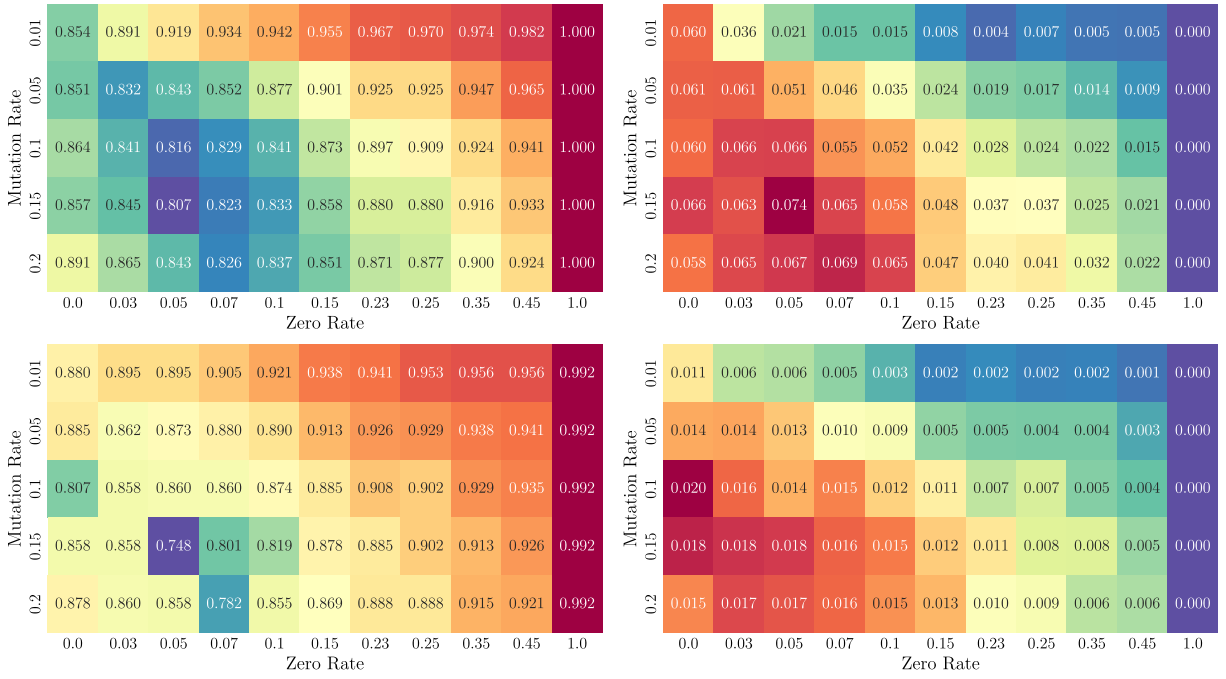


Fig. 6. Hyper-parameter study for `Sonic` focusing on mutation probability (p_m) and zero probability (p_z) used in the mutation operator (presented in Section 3.3). The analysis is conducted on 20 Newsgroups (top row) with $s = 0.1$ and $\delta = 0.5$, and MNIST (bottom row) with $s = 0.05$ and $\delta = 0.2$. The left column displays AMI scores, while the right column shows the values of the cost function from Eq. (1). The crossover probability (p_c) is fixed at 0.85.

increase the number of iterations needed to achieve optimal results. Overall, the hyperparameters chosen in Section 4.1 provide a balanced trade-off between stealthiness and effectiveness. This balance can be adjusted in practical scenarios according to the attacker's specific goals.

5. Related works

The focus on robustness and adversarial machine learning has traditionally been on supervised learning (e.g., [13] or [15]), leaving unsupervised learning comparatively under-explored in this context. Initial contributions in this area were made by Dutrisac and Skillicorn [19,43], followed by a more comprehensive investigation by Biggio et al. [5], who proposed a categorization and theoretical framework for the challenge of adversarial clustering. Additionally, the same work [5] introduced a perfect knowledge attack targeting single-linkage clustering and, in a subsequent study, extended this to complete linkage hierarchical clustering [4], particularly within the computer security domain. Crussell and Kegelmeyer [16] devised an attack on DBSCAN [22] by exploiting the inherent vulnerability of density-based clustering algorithms, creating bridges between clusters. Chhabra et al. [11] relaxed the assumptions of perfect knowledge made by previous works by developing a derivative-free strategy that perturbs a single data point in a linearly separable task without requiring prior knowledge of the clustering algorithm's metric. This threat model is further addressed

by Cinà et al. [14], who developed a black-box poisoning attack to test the robustness of clustering algorithms. Their approach requires no knowledge of the clustering algorithm or its parameters and allows for the perturbation of multiple data points simultaneously. Further work has been conducted by Xu et al. [48], where they propose an adversarial attack to fool subspace clustering. It achieves misclassification by applying adversarial manipulations inside the linear subspace to move a sample toward the target class. In a recent study by Zhang and Tang [50], a new data poisoning attack was introduced, targeting deep clustering models [41] , exploiting the robust features of clustering categories. Additionally, the study discusses creating adversarial examples against K-means and Gaussian mixture models by manipulating clean input to the decision boundary.

Overall scalability remains a primary concern when dealing with attacks against clustering algorithms [5,14], especially density and hierarchy-based. This challenge either slows down the algorithms or forces them to adopt heuristics to maintain efficiency. `Sonic` addresses these issues by combining efficiency and efficacy, providing fast benchmarking of unsupervised systems while ensuring the quality of the resulting perturbations.

6. Conclusions, limitations, and future works

This work proposes `Sonic`, a genetic optimization poisoning attack against clustering algorithms. `Sonic` speeds up the poisoning process by acknowledging two main insights: (i) during practical poisoning attacks, only a small subset of data is tampered with by the attacker, and (ii) most of the clustering operations (e.g., distances between samples) on clean data points do not need to be re-computed. To this end, `Sonic` leverages a surrogate incremental clustering model, i.e., FISHDBC, to mitigate the scalability problems of previous state-of-the-art iterative methods, removing the burden of re-clustering the whole dataset at each optimization step. We report an experimental evaluation spanning four different datasets, both in the image and text domains, showing the effectiveness and efficiency of the attack. `Sonic` successfully disrupts target clustering algorithms, achieving comparable performance to attacks that directly incorporate the target algorithm in the optimization process at a fraction of the execution time.

Furthermore, we demonstrate that attacks generated with `Sonic` can effectively transfer to other clustering algorithms such as HDBSCAN*, DBSCAN*, and hierarchical single linkage. However, methods such as complete and average linkage clustering exhibit greater resilience, underscoring the need for algorithm-specific attack strategies to better evaluate and enhance the robustness of clustering techniques. A primary limitation of `Sonic` lies in its reliance on incremental surrogate clustering algorithms, which are required to speed up its poisoning sample optimization. Incremental algorithms enable `Sonic` to efficiently adapt clustering outcomes in response to small changes, significantly reducing computational overhead compared to re-clustering the entire dataset. However, such algorithms are not universally available across all clustering methods, restricting `Sonic`'s applicability in specific scenarios. Addressing this limitation could involve designing additional incremental algorithms tailored to different clustering techniques. Furthermore, `Sonic` currently employs genetic algorithms to optimize adversarial perturbations. While genetic algorithms are robust and versatile [23], they are generally less efficient than other iterative optimization methods. Future work could explore integrating more computationally efficient optimization strategies, such as Particle Swarm Optimization [28]. In conclusion, `Sonic` represents a significant advancement in the robustness verification of clustering algorithms, enabling faster, and reliable, benchmarking of unsupervised systems, even as dataset sizes continue to grow.

CRediT authorship contribution statement

Francesco Villani: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Dario Lazzaro:** Writing – review & editing, Visualization, Validation, Software, Methodology, Investigation, Data curation. **Antonio Emanuele Cinà:** Writing – review & editing, Writing – original draft, Validation, Software, Supervision, Project administration, Methodology, Conceptualization, Investigation, Formal analysis. **Matteo Dell'Amico:** Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Formal analysis, Conceptualization. **Battista Biggio:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Funding acquisition. **Fabio Roli:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by project FISA-2023–00128 funded by the MUR program "Fondo italiano per le scienze applicate"; the European Union's Horizon Europe research and innovation program under the project ELSA, grant agreement No 101070617; by Fondazione di Sardegna under the project "TrustML: Towards Machine Learning that Humans Can Trust", CUP: F73C22001320007; by EU - NGEU National Sustainable Mobility Center (CN00000023) Italian Ministry of University and Research Decree n. 1033—17/06/2022 (Spoke 10); and by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU. Lastly, this work was carried out while Dario Lazzaro was enrolled in the Italian National Doctorate on Artificial Intelligence run by the Sapienza University of Rome in collaboration with the University of Genoa, funded by "Unione europea-Next Generation EU, Missione 4 Componente 1 CUP B53C23003580004".

Data availability

We utilize a publicly known dataset. The source code is available at https://github.com/francescovillani/sonic-poisoning-clustering.

References

[1] B. Acharya, D. Lazzaro, E. López-Morales, A. Oest, M. Saad, A.E. Cinà, L. Schönherr, T. Holz, The imitation game: exploring brand impersonation attacks on social media platforms, in: 33rd USENIX Security Symposium (USENIX Security 24), 2024, pp. 4427–4444.

[2] K. Albishre, M. Albathan, Y. Li, Effective 20 newsgroups dataset cleaning, in: 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT), IEEE, 2015, pp. 98–101.

[3] M. Ankerst, M.M. Breunig, H.-P. Kriegel, J. Sander, Optics: ordering points to identify the clustering structure, ACM Sigmod Rec. 28 (1999) 49–60.

[4] B. Biggio, S.R. Bulò, I. Pillai, M. Mura, E.Z. Mequanint, M. Pelillo, F. Roli, Poisoning complete-linkage hierarchical clustering, in: Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshop, S+ SSPR, Springer, 2014, pp. 42–52.

[5] B. Biggio, I. Pillai, S. Rota Bulò, D. Ariu, M. Pelillo, F. Roli, Is data clustering in adversarial settings secure? in: Proceedings of the 2013 ACM Workshop on Artificial Intelligence and Security, 2013, pp. 87–98.

[6] M. Binois, N. Wycoff, A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization, ACM Trans. Evol. Learn. Optim. 2 (2022) 1–26.

[7] R.J.G.B. Campello, D. Moulavi, J. Sander, Density-based clustering based on hierarchical density estimates, in: J. Pei, V.S. Tseng, L. Cao, H. Motoda, G. Xu (Eds.), Advances in Knowledge Discovery and Data Mining, Berlin, Heidelberg, Springer Berlin Heidelberg, 2013, pp. 160–172.

[8] N. Carlini, M. Jagielski, C.A. Choquette-Choo, D. Paleka, W. Pearce, H. Anderson, A. Terzis, K. Thomas, F. Tramèr, Poisoning web-scale training datasets is practical, arXiv preprint arXiv:2302.10149, 2023.

[9] Y. Chang, Z. Tu, W. Xie, B. Luo, S. Zhang, H. Sui, J. Yuan, Video anomaly detection with spatio-temporal dissociation, Pattern Recognit. 122 (2022) 108213.

[10] H. Chen, M. Liang, W. Liu, W. Wang, P.X. Liu, An approach to boundary detection for 3d point clouds based on dbscan clustering, Pattern Recognit. 124 (2022) 108431.

[11] A. Chhabra, A. Roy, P. Mohapatra, Suspicion-free adversarial attacks on clustering algorithms, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 3625–3632.

[12] A.E. Cinà, K. Grosse, A. Demontis, S. Vascon, W. Zellinger, B.A. Moser, A. Oprea, B. Biggio, M. Pelillo, F. Roli, Wild patterns reloaded: a survey of machine learning security against training data poisoning, ACM Comput. Surv. 55 (2023) 1–39.

[13] A.E. Cinà, J. Rony, M. Pintor, L. Demetrio, A. Demontis, B. Biggio, I.B. Ayed, F. Roli, Attackbench: evaluating gradient-based attacks for adversarial examples, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2025a, pp. 2600–2608, https://doi.org/10.1609/aaai.v39i3.32263

[14] A.E. Cinà, A. Torcinovich, M. Pelillo, A black-box adversarial attack for poisoning clustering, Pattern Recognit. 122 (2022) 108306, https://doi.org/10.1016/j.patcog.2021.108306

[15] A.E. Cinà, F. Villani, M. Pintor, L. Schönherr, B. Biggio, M. Pelillo, σ-zero: gradient-based optimization of ℓ_0-norm adversarial examples, in: The Thirteenth International Conference on Learning Representations, 2025b, https://openreview.net/forum?id=JMPOqoe4tl.

[16] J. Crussell, P. Kegelmeyer, Attacking dbscan for fun and profit, in: Proceedings of the 2015 SIAM International Conference on Data Mining, SIAM, 2015, pp. 235–243.

[17] M. Dell'Amico, FISHDBC: Flexible, incremental, scalable, hierarchical density-based clustering for arbitrary data and distance, arXiv preprint arXiv:1910.07283, 2019.

[18] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, F. Roli, Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks, in: 28th USENIX Security Symposium (USENIX Security 19), 2019, pp. 321–338.

[19] J.G. Dutrisac, D.B. Skillicorn, Hiding clusters in adversarial settings, in: 2008 IEEE International Conference on Intelligence and Security Informatics, IEEE, 2008, pp. 185–187.

[20] A.E. Eiben, E.H.L. Aarts, K.M. Van Hee, Global convergence of genetic algorithms: a markov chain analysis, in: Parallel Problem Solving from Nature: 1st Workshop, PPSN I Dortmund, Springer, 1991, pp. 3–12.

[21] M. El Yabroudi, K. Awedat, R.C. Chabaan, O. Abudayyeh, I. Abdel-Qader, Adaptive dbscan lidar point cloud clustering for autonomous driving applications, in: 2022 IEEE International Conference on Electro Information Technology (eIT), IEEE, 2022, pp. 221–224.

[22] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: KDD, 1996, pp. 226–231.

[23] K. Gallagher, M. Sambridge, Genetic algorithms: a powerful tool for large-scale nonlinear optimization problems, Comput. Geosci. 20 (1994) 1229–1236.

[24] J. Gan, Y. Tao, Dbscan revisited: mis-claim, un-fixability, and approximation, in: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, Association for Computing Machinery, 2015, pp. 519–530, https://doi.org/10.1145/2723372.2737792

[25] A. Ghosal, A. Nandy, A.K. Das, S. Goswami, M. Panday, A short review on different clustering techniques and their applications, Emerg. Technol. Modelling Graph. Proc. IEM Graph 2018 (2020) 69–83.

[26] M. Grootendorst, Bertopic: Neural topic modeling with a class-based tf-idf procedure, arXiv preprint arXiv:2203.05794, 2022.

[27] L. Hubert, P. Arabie, Comparing partitions, J. Classif. 2 (1985) 193–218.

[28] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proceedings of ICNN'95-International Conference on Neural Networks, Ieee, 1995, pp. 1942–1948.

[29] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.

[30] K. Lang, Newsweeder: learning to filter netnews, in: Proceedings of the Twelfth International Conference on Machine Learning, 1995, pp. 331–339.

[31] Y. LeCun, C. Cortes, The mnist database of handwritten digits, 2005.

[32] Y.A. Malkov, D.A. Yashunin, Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2020) 824–836.

[33] L. McInnes, J. Healy, Accelerated hierarchical density based clustering, in: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 2017, pp. 33–42, https://doi.org/10.1109/ICDMW.2017.12

[34] L. McInnes, J. Healy, S. Astels, et al., Hdbscan: hierarchical density based clustering, J. Open Source Softw. 2 (2017) 205.

[35] L. McInnes, J. Healy, J. Melville, Umap: Uniform manifold approximation and projection for dimension reduction, arXiv:1802.03426, 2020.

[36] M. Meilă, Comparing clusterings—an information based distance, J. Multivar. Anal. 98 (2007) 873–895.

[37] X.V. Nguyen, J. Epps, J. Bailey, Information theoretic measures for clusterings comparison: is a correction for chance necessary? in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009, 2009, pp. 1073–1080.

[38] K. Pearson, VII. Note on regression and inheritance in the case of two parents, Proc. R. Soc. Lond. 58 (1895) 240–242.

[39] M.A. Rahat, V. Banerjee, G. Bloom, Y. Zhuang, Cimalir: cross-platform IOT malware clustering using intermediate representation, in: 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2024, pp. 0460–0466.

[40] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv:1908.10084, 2019.

[41] Y. Ren, J. Pu, Z. Yang, J. Xu, G. Li, X. Pu, P.S. Yu, L. He, Deep clustering: a comprehensive survey, IEEE Trans. Neural Netw. Learn. Syst. 36 (2024) 5858–5878.

[42] V. Shejwalkar, A. Houmansadr, P. Kairouz, D. Ramage, Back to the drawing board: a critical evaluation of poisoning attacks on production federated learning, in: 2022 IEEE Symposium on Security and Privacy (SP), IEEE, 2022, pp. 1354–1371.

[43] D.B. Skillicorn, Adversarial knowledge discovery, IEEE Intell. Syst. 24 (2009) 54.

[44] P.H.A. Sneath, Numerical taxonomy, in: Bergey's Manual® of Systematic Bacteriology, Springer, 2005, pp. 39–42.

[45] C. Spearman, The proof and measurement of association between two things, 1961.
[46] J.H. Ward Jr, Hierarchical grouping to optimize an objective function, J. Am. Stat. Assoc. 58 (1963) 236–244.
[47] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, arXiv preprint arXiv:1708.07747, 2017.
[48] Y. Xu, X. Wei, P. Dai, X. Cao, A2sc: adversarial attacks on subspace clustering, ACM Trans. Multimed. Comput. Commun. Appl. 19 (2023) 1–23.
[49] H. Yin, Z. Zhang, Z. Wang, Y. Özyurt, W. Liang, W. Dong, Y. Zhao, Y. Shan, Behavioral graph fraud detection in e-commerce, in: 2022 IEEE International Conference on Data Mining Workshops (ICDMW), IEEE, 2022, pp. 1–8.
[50] C. Zhang, Z. Tang, Novel poisoning attacks for clustering methods via robust feature generation, Neurocomputing (2024) 127925.